Google Scholar BETA

suspend mutator code at a safe point + set pro Search

Advanced Scholar Search
Scholar Preferences
Scholar Help

The following words are very common and were not included in your search: at a. [details]

Scholar Results **1 - 10** of about **45** for **suspend mutator code at a safe point + set processor state**. (0.

A Fast Analysis for Thread-Local Garbage Collection with Dynamic Class Loading
R Jones, AC King - cs.kent.ac.uk
... resources between threads, it is no longer necessary to **suspend** all **mutator** threads because ... the-world phase in which spe- cialisation and **code** patching is ...
View as HTML - Web Search

An on-the-fly mark and sweep garbage collector based on sliding views
H Azatchi, Y Levanoni, H Paz, E Petrank - OOPSLA'03 ACM Conference on Object-Oriented Systems, ... - portal.acm.org
... Fourth handshake: during this handshake each **mutator** is stopped ... 3.4 Collector **code** Collector's **code** for cycle k is ... for each thread T do 5. **suspend** thread T 6 ...
Cited by 12 - cs.technion.ac.il - cs.purdue.edu - portal.acm.org

An On-the-Fly Reference Counting Garbage Collector for Java
Y Levanoni, E Petrank - OOPSLA, 2001 - portal.acm.org
... that the garbage collector thread may **suspend** and subsequently ... intermediate algorithm together with full **code** is given ... Each **mutator** at a time will provide its ...
Cited by 21 - rpi.edu - cs.technion.ac.il - portal.acm.org

The Jalapeno virtual machine
B Alpern, CR Attanasio, JJ Barton, MG Burke, P ... - IBM Systems Journal, 2000 - cs.ucsb.edu
... This allows compilers to optimize **code** (by maintaining an internal pointer, for example) be- tween **safe** points that ... If the thick bit is not **set**, the rest of ...
Cited by 230 - View as HTML - Web Search - stanford.edu - cs.anu.edu.au - research.ibm.com - all 10 versions »

Very Concurrent Mark-&-Sweep Garbage Collection without Fine-Grain Synchronization
L Huelsbergen, P Winterbottom, L Technologies - ISMM, 1998 - portal.acm.org
... tributed operating system that supports mobile **code**. ... (A **suspend**-thread **mutator** ... At this **point** the **mutator** thread is suspended and the mu- tator's root **set** is ...
Cited by 30 - Web Search - cm.bell-labs.com - cs.bell-labs.com - portal.acm.org

Age-oriented garbage collection
H Paz, E Petrank, SM Blackburn - 2003 - cs.technion.ac.il
... A typical such information is a list of inter-generational **point**- ers. ... CurrPos := TempPos 12. // **set** dirty 13. ... Fig. 1. **Mutator code**: Update Operation ...
Cited by 3 - View as HTML - Web Search - cs.technion.ac.il

An On-the-Fly Reference-Counting Garbage Collector for Java

E PETRANK - cs.technion.ac.il
... dependent), and (b) no protected **code** is being ... is assumed that the collector thread may **suspend** and subsequently ... When a **mutator** is suspended, the collector may ...
View as HTML - cs.technion.ac.il

GC points in a threaded environment
O Agesen - 1998 - sunlabs.com
... a GC trap will cause the thread to **suspend** itself. ... **safe** to re-execute them when the **mutator** resumes after ... into the just-in-time compiler's **code** generator was ...
Cited by 20 - View as HTML - Web Search - research.sun.com - sun.com - all 8 versions » - Library Search

Lorenz Huelsbergen
L Technologies - cs.bell-labs.com
... dis- tributed operating system that supports mobile **code**. ... data have COLOR(epoch) */ (j) **suspend** thread **mutator** ... At this **point** the **mutator** thread is suspended ...
View as HTML - Web Search - cs.bell-labs.com - netlib.bell-labs.com - cm.bell-labs.com - all 5 versions »

[PS] Supporting Type-**Safe** Languages on DSM Systems
W Yu - cs.rice.edu
... Supporting Type-**Safe** Languages on DSM Systems ... inating the need for the programmer to write **code** to track ... run-time type information is necessary to **set** up the ...
View as HTML - Web Search

Gooooogle ▶
Result Page: 1 2 3 4 5 **Next**

Published since January 1990 and Published before October 2001 Found 47 of 74

Terms used **mutator safe point set state**

Sort results by relevance Save results to a Binder Try an Advanced Search
Search Tips Try this search in The ACM Guide
Display results expanded form Open results in a new window

Results 1 - 20 of 47 Result page: **1** 2 3 next

Relevance scale

1 Using shape analysis to reduce finite-state models of concurrent Java programs
James C. Corbett
January 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 9 Issue 1
Full text available: pdf(284.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Finite-state verification (e.g., model checking) provides a powerful means to detect concurrency errors, which are often subtle and difficult to reproduce. Nevertheless, widespread use of this technology by developers is unlikely until tools provide automated support for extracting the required finite-state models directly from program source. Unfortunately, the dynamic features of modern languages such as Java complicate the construction of compact finite-state models for verification. I ...

Keywords: Java, concurrent systems, finite-state verification, model extraction, modeling, shape analysis, state-space reductions

2 Type-preserving garbage collectors
Daniel C. Wang, Andrew W. Appel
January 2001 **ACM SIGPLAN Notices , Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 36 Issue 3
Full text available: pdf(221.47 KB) Additional Information: full citation, abstract, references, citings, index terms

By combining existing type systems with standard type-based compilation techniques, we describe how to write strongly typed programs that include a function that acts as a t racing garbage collector for the program. Since the garbage collector is an explicit function, we do not need to provide a t rusted garbage collector as a runtime service to manage memory.Since our language is strongly typed, the standard type soundness guarantee "Well typed programs do not gc wrong" is extended to include t ...

3 Polymorphic splitting: an effective polyvariant flow analysis
Andrew K. Wright, Suresh Jagannathan
January 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 20 Issue 1
Full text available: pdf(517.76 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article describes a general-purpose program analysis that computes global control-flow and data-flow information for higher-order, call-by-value languages. The analysis employs a novel form of polyvariance called polymorhic splitting that uses let-expressions as syntactic clues to gain precision. The information derived from the analysis is used both to eliminate run-time checks and to inline procedure. The analysis and optimizations have been applied to a suite of Scheme progra ...

Keywords: flow analysis, inlining, polyvariance, run-time checks

4 Principled scavenging
Stefan Monnier, Bratin Saha, Zhong Shao
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

Proof-carrying code and typed assembly languages aim to minimize the trusted computing base by directly certifying the actual machine code. Unfortunately, these systems cannot get rid of the dependency on a trusted garbage collector. Indeed, constructing a provably type-safe garbage collector is one of the major open problems in the area of certifying compilation.

Building on an idea by Wang and Appel, we present a series of new techniques for writing type-safe stop-and-copy garbage c ...

5 Sapphire: copying GC without stopping the world
Richard L. Hudson, J. Eliot B. Moss
June 2001 **Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande**
Full text available: pdf(899.45 KB) Additional Information: full citation, abstract, references, citings, index terms

Many concurrent garbage collection (GC) algorithms have been devised, but few have been implemented and evaluated, particularly for the Java programming language. Sapphire is an algorithm we have devised for concurrent copying GC. Sapphire stresses minimizing the amount of time any given application thread may need to block to support the collector. In particular, Sapphire is intended to work well in the presence of a large number of application threads, on small- to medium-scale shared memor ...

6 Borrow, copy or steal?: loans and larceny in the orthodox canonical form
Anthony J. H. Simons
October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 33 Issue 10
Full text available: pdf(2.09 Additional Information: full citation, abstract, references, index

MB) terms

Dynamic memory management in C++ is complex, especially across the boundaries of library abstract data types. C++ libraries designed in the orthodox canonical form (OCF) alleviate some of the problems by ensuring that classes which manage any kind of heap structures faithfully copy and delete these. However, in certain common circumstances, OCF heap structures are wastefully copied multiple times. General reference counting is not an option in OCF, since a shared body violates the intended value ...

Keywords: C++, borrowing, copy-on-write, implementation strategies, larceny, memory management, stealing, transfer of ownership

7 Garbage collecting the Internet: a survey of distributed garbage collection
Saleh E. Abdullahi, Graem A. Ringwood
September 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 3
Full text available: pdf(337.65 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Internet programming languages such as Java present new challenges to garbage-collection design. The spectrum of garbage-collection schema for linked structures distributed over a network are reviewed here. Distributed garbage collectors are classified first because they evolved from single-address-space collectors. This taxonomy is used as a framework to explore distribution issues: locality of action, communication overhead and indeterministic communication latency.

Keywords: automatic storage reclamation, distributed, distributed file systems, distributed memories, distributed object-oriented management, memory management, network communication, object-oriented databases, reference counting

8 Very concurrent mark-&-sweep garbage collection without fine-grain synchronization
Lorenz Huelsbergen, Phil Winterbottom
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management**, Volume 34 Issue 3
Full text available: pdf(1.36 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a new incremental algorithm for the concurrent reclamation of a program's allocated, yet unreachable, data. Our algorithm is a variant of mark-&-sweep collection that---unlike prior designs---runs mutator, marker, and sweeper threads concurrently *without* explicit fine-grain synchronization on shared-memory multiprocessors. A global, but infrequent, synchronization coordinates the per-object coloring marks used by the three threads; fine-grain synchronization is achieve ...

9 An on-the-fly reference counting garbage collector for Java
Yossi Levanoni, Erez Petrank
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications**, Volume 36 Issue 11
Full text available: pdf(280.30 Additional Information: full citation, abstract, references, citings,

KB) index terms

Reference counting is not naturally suitable for running on multiprocessors. The update of pointers and reference counts requires atomic and synchronized operations. We present a novel reference counting algorithm suitable for a multiprocessor that does not require any synchronized operation in its write barrier (not even a compare-and-swap type of synchronization). The algorithm is efficient and may complete with any tracing algorithm.

10 Java without the coffee breaks: a nonintrusive multiprocessor garbage collector
David F. Bacon, Clement R. Attanasio, Han B. Lee, V. T. Rajan, Stephen Smith
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5
Full text available: pdf(1.69 MB) Additional Information: full citation, abstract, references, citings, index terms

The deployment of Java as a concurrent programming language has created a critical need for high-performance, concurrent, and incremental multiprocessor garbage collection. We present the *Recycler*, a fully concurrent pure reference counting garbage collector that we have implemented in the Jalapeño Java virtual machine running on shared memory multiprocessors.

While a variety of multiprocessor collectors have been proposed and some have been implemented, experimental dat ...

11 Collecting distributed garbage cycles by back tracing
Umesh Maheshwari, Barbara Liskov
August 1997 **Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing**
Full text available: pdf(1.21 MB) Additional Information: full citation, references, citings, index terms

12 Barrier techniques for incremental tracing
Pekka P. Pirinen
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management**, Volume 34 Issue 3
Full text available: pdf(707.56 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a classification of barrier techniques for interleaving tracing with mutator operation during an incremental garbage collection. The two useful tricolour invariants are derivec from more elementary considerations of graph traversal. Barrier techniques for maintaining these invariants are classified according to the action taken at the barrier (such as scanning an object or changing its colour), and it is shown that the algorithms described in the literature cover all the poss ...

13 A practical soft type system for scheme
Andrew K. Wright, Robert Cartwright
January 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 1
Full text available: pdf(624.50 Additional Information: full citation, abstract, references, citings,

KB) index terms

A soft type system infers types for the procedures and data structures of dynamically typed programs. Like conventional static types, soft types express program invariants and thereby provide valuable information for program optimization and debugging. A soft type checker uses the types inferred by a soft type system to eliminate run-time checks that are provably unnecessary; any remaining run-time checks are flagged as potential program errors. Sof ...

Keywords: run-time checks, soft typing

14 Implementing jalapeño in Java
Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeño started, a boot image of a ...

15 A concurrent, generational garbage collector for a multithreaded implementation of ML
Damien Doligez, Xavier Leroy
March 1993 **Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the design and implementation of a "quasi real-time" garbage collector for Concurrent Caml Light, an implementation of ML with threads. This two-generation system combines a fast, asynchronous copying collector on the young generation with a non-disruptive concurrent marking collector on the old generation. This design crucially relies on the ML compile-time distinction between mutable and immutable objects.

16 Scheduling garbage collector for embedded real-time systems
Taehyoun Kim, Naehyuck Chang, Namyun Kim, Heonshik Shin
May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 workshop on Languages, compilers, and tools for embedded systems**, Volume 34 Issue 7
Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, index terms

This paper proposes a new scheduling method for multiple mutators and a garbage collector running on embedded real-time systems with a single processor and no virtual memory. The hard real-time tasks should reserve a certain amount of heap memory to prevent memory starvation and/or deadline miss. Since the memory requirement depends on the worst-case response time of a garbage collector, the traditional approach in which garbage collection is performed in the background demands large memory spac ...

17 Cycles to recycle: garbage collection to the IA-64
Richard L. Hudson, J. Elliot Moss, Sreenivas Subramoney, Weldon Washburn
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, citings, index terms

The IA-64, Intel's 64-bit instruction set architecture, exhibits a number of interesting architectural features. Here we consider those features as they relate to supporting garbage collection (GC). We aim to assist GC and compiler implementors by describing how one may exploit features of the IA-64. Along the way, we record some previously unpublished object scanning techniques, and offer novel ones for object allocation (suggesting some simple operating system support that would simplify it ...

18 A parallel, real-time garbage collector
Perry Cheng, Guy E. Blelloch
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5
Full text available: pdf(1.82 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a parallel, real-time garbage collector and present experimental results that demonstrate good scalability and good real-time bounds. The collector is designed for shared-memory multiprocessors and is based on an earlier collector algorithm [2], which provided fixed bounds on the time any thread must pause for collection. However, since our earlier algorithm was designed for simple analysis, it had some impractical features. This paper presents the extensions necessary for a pract ...

19 In-place updates in the presence of control operators
Sandip K. Biswas
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming**, Volume VII Issue 3
Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, index terms

This paper presents a formal account of the concept of in-place updates in purely functional languages. In purely functional languages, updates of abstract objects involve creating duplicates of these objects. This paper reviews static conditions, which, if satisfied by &lgr;-terms, guarantee that, even if updates are performed in-place, the purely functional semantics is retained. These static conditions, however, fail to guarantee the requisite safety in the presence of control operators ...

20 Automatic inline allocation of objects
Julian Dolby
May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation**, Volume 32 Issue 5
Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented languages like Java and Smalltalk provide a uniform object model that simplifies

programming by providing a consistent, abstract model of object behavior. But direct implementations introduce overhead, removal of which requires aggressive implementation techniques (e.g. type inference, function specialization); in this paper, we introduce *object inlining*, an optimization that automatically inline allocates objects within containers (as is done by hand in C++) within a unif ...

Results 1 - 20 of 47 Result page: **1** 2 3 next

Results for "(((mutator safe point set state suspension suspend)<in>metadata)) <and> (pyr >= 1990 ..."
Your search matched **0** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

(((mutator safe point set state suspension suspend)<in>metadata)) <and:

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages assistance revising your search.

Search Results

Results for "**(((mutator safe point)<in>metadata)) <and> (pyr >= 1990 <and> pyr <= 2001)**" e-mail
Your search matched **0** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History
New Search

Modify Search

(((mutator safe point)<in>metadata)) <and> (pyr >= 1990 <and> pyr <= 2

☐ Check to search only within this results set

Display Format: ◉ Citation ○ Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages assistance revising your search.

Welcome United States Patent and Trademark Office

Search Results BROWSE SEARCH IEEE XPLORE GUIDE

Results for "(((garage collection safe point)<in>metadata)) <and> (pyr >= 1990 <and> pyr &l..." e-mail
Your search matched 0 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by Relevance in Descending order.

» Search Options

View Session History
New Search

Modify Search

(((garage collection safe point)<in>metadata)) <and> (pyr >= 1990 <and>

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages assistance revising your search.

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	636	717/124	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 06:46
S2	2	"6308319".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/07 15:37
S3	3	("6308319").URPN.	USPAT	OR	OFF	2004/12/07 15:37
S4	7	("5088036" | "5321834" | "5560003" | "5920876" | "5953736" | "6098089" | "6101580").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2004/12/07 16:07
S5	0	(garbage adj collection) or (heap near3 management)) and deallocat$3 near5 (heuristic or strateg$3 or procedure) same pointer and (multi-thread$3 or parallel$7) and ((vliw or epic) near5 (slot or pipelin$3) and (halt$3 or suspend$4 or resum$3 or resumption)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/07 16:31
S6	0	("6618738").URPN.	USPAT	OR	OFF	2004/12/07 16:19
S7	8	("20010000821" | "20020120823" | "5953736" | "6052699" | "6065020" | "6308319" | "6393440" | "6427154").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2004/12/07 16:19
S8	6	("6467075").URPN.	USPAT	OR	OFF	2004/12/07 16:23
S9	12	("5703789" | "5764951" | "5903466" | "5930827" | "6021132" | "6044418" | "6067608" | "6076151" | "6253226" | "6263302" | "6295594" | "6308319").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2004/12/07 16:26
S10	4	("6427154").URPN.	USPAT	OR	OFF	2004/12/07 16:28
S11	0	((garbage adj collection) or (heap near3 management)) and (deallocat$3 near5 (heuristic or strateg$3 or procedure)) and pointer and (multi-thread$3 or parallel$7) and ((vliw or epic) near5 (slot or pipelin$3)) and (halt$3 or suspend$4 or resum$3 or resumption)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/07 16:33

S12	0	((garbage adj collection) or (heap near3 manag$5)) and ((deallocat$3 or de-allocat$3) near5 (heuristic or strateg$3 or procedure)) and pointer and (multi-thread$3 or parallel$7) and (vliw or epic) and (slot or pipelin$3) and (halt$3 or suspend$4 or resum$3 or resumption)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/07 16:34
S13	5	((garbage adj collect$3) or (heap near3 manag$5)) and (deallocat$3 or de-allocat$3) and (heuristic or strateg$3 or procedure) and pointer and (multi-thread$3 or parallel$7 or multithread$3) and (vliw or epic) and (slot or pipelin$3) and (halt$3 or suspend$4 or resum$3 or resumption)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/07 16:51
S14	2	"20020052926"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/07 16:52
S15	189	(instruction adj (encod$3 or arrangement)) and (VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:23
S16	257	((instruction or word) adj (encod$3 or arrangement)) and (VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:40
S17	4212	(nop or noop or no-op) and (optimiz$5 or remov$3 or delet$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:23
S18	112	S16 and S17	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 07:09
S19	91	(nop or noop or no-op) near5 (optimiz$5 or remov$3 or delet$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 07:38

S20	3	S16 and S19	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 07:09
S21	225	(nop or noop or no-op or no-operation or (no adj operation)) near5 (optimiz$5 or remov$3 or delet$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 07:42
S22	295	(nop or noop or no-op or no-operation or (no adj operation)) near5 (replac$3 or substitut$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 07:48
S23	7	S16 and S22	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 07:43
S24	175	(replac$3 or substitut$3) adj (nop or noop or no-op or no-operation or (no adj operation))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:23
S25	6	("6195756").URPN.	USPAT	OR	OFF	2004/12/08 08:00
S26	1	encod$3 near3 (unused adj position)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:24
S27	7	encod$3 near3 ((unused or available) adj (position or slot))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:26
S28	100	encod$3 near3 ((unused or available) near3 (position or slot))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:30
S29	1	encod$3 near3 ((unused or available) near3 (position or slot)) and (mmu or garbage or gc or heap)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:27
S30	0	compiler near5 (encod$3 near5 heuristic)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:31

S31	55	(encod$3 near5 heuristic)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:31
S32	0	((instruction or word) adj (encod$3 or arrangement)) same heuristic and (VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 08:40
S33	0	((instruction or word) adj (encod$3 or arrangement or placement)) same heuristic and (VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 09:00
S34	40	(exploit$3 or insert$3) same (heap or mmu or garbage or gc) same ((horizontal$2 near3 instruction) or word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 13:21
S35	231	(exploit$3 or insert$3 or add$3 or modif$7) same (heap or mmu or garbage or gc) same ((horizontal$2 near3 instruction) or word or VLIW or EPIC or SIMD or MIMD or (explicit$2 adj parallel$5))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 14:29
S36	33	(safe adj point) same (suspend$3 or mutator or exception or event)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 14:31
S37	20	((safe adj point) same (suspend$3 or mutator or exception or event)) not sun.as.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 14:50
S38	1034	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator) and (word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 17:06

S39	155	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator) same (word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 14:54
S40	27	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator) near5 (word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 14:54
S41	29	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator or trap$4) near5 (word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 14:55
S42	38	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator or trap$4) near7 (word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 15:18
S43	25	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator or trap$4) near7 ((instruction adj word) or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay or super-scaler or superscaler or (encod$3 near3 horizontal$2) or (plurality near2 (primitive or operations)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/08 15:21
S44	5	("5557771").URPN.	USPAT	OR	OFF	2004/12/08 16:26

S45	15	(instruction near3 patch$3) and (suspend$3 or suspension or stall$3 or interrupt or trap$3 or safe or (context adj switch$3) or (high$3 near2 priority)) and (garbage or heap or mmu or (memory adj manag$4)) and ((instruction adj word) or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay or super-scaler or superscaler or (encod$3 near3 horizontal$2) or (plurality near2 (primitive or operations)))	USPAT	OR	OFF	2004/12/08 16:00
S46	284	(suspend$3 or suspension or stall$3 or interrupt or trap$3 or safe or (context adj switch$3) or (high$3 near2 priority)) and (garbage or heap or mmu or (memory adj manag$4)) same ((instruction adj word) or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay or super-scaler or superscaler or (encod$3 near3 horizontal$2) or (plurality near2 (primitive or operations)))	USPAT	OR	OFF	2004/12/08 16:04
S47	54	(suspend$3 or suspension or stall$3 or interrupt or trap$3 or safe or (context adj switch$3) or (high$3 near2 priority)) same (garbage or heap or mmu or (memory adj manag$4)) same ((instruction adj word) or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay or super-scaler or superscaler or (encod$3 near3 horizontal$2) or (plurality near2 (primitive or operations)))	USPAT	OR	OFF	2004/12/08 16:04
S48	1	("5557771").pn.	USPAT	OR	OFF	2004/12/08 16:31
S49	0	("5557771").pn. and trap$4	USPAT	OR	OFF	2004/12/08 16:31
S50	1	(garbage or heap or mmu or (memory adj manag$4)) same (interrupt$3 or suspen$4 or (context adj switch) or (priority near3 thread) or stall$3 or mutator) and (augment$3 near3 (word or VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel) or slot or delay))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/08 17:07

S51	387	safe adj point	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 10:14
S52	7	(safe adj point) near3 garbage	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 10:15
S53	0	(saf$2 adj interrupt$3) near3 garbage	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 10:15
S54	0	(saf$2 adj interrupt$3) near3 compiler	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 10:15
S55	1	(saf$2 adj interrupt$3) near3 thread	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:22
S56	704	packed near3 instruction	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:49
S57	191	(instruction adj (encod$3 or arrangement)) and (VLIW or EPIC or SIMD or MIMD or (explicitly adj parallel))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 15:11
S58	4214	(nop or noop or no-op) and (optimiz$5 or remov$3 or delet$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:24
S59	22	S56 and S57	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:24
S60	142	S56 and S58	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:24

S61	5	S56 and S58 and S57	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 12:24
S62	15	packed near3 instruction and (unused near3 (operat$3 or instruction or position or slot))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 15:05
S63	2	"20020052926"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 14:41
S64	2	"6308319".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 14:56
S65	4	("6151668" "5832205").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 14:57
S66	60	packed near3 (instruction or format or word) and (unused near3 (operat$3 or instruction or position or slot))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 15:06
S67	45	S66 not S62	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/12/09 15:06
S68	3	(conditional adj trap) and JIT	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/13 09:40
S69	3	(conditional adj trap) and JIT	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/13 09:46
S70	14	((exception or trap$4) near5 conditional) and JIT	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/12/13 09:46

S71	2	"6308319".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/01 11:59
S72	2	"20020052926"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 07:27
S73	167	((explicitly near3 parallel$3) or epic or vliw) and (breakpoint or commit) and (status or bit or value) and (encod$3 or "unused position" or "delay slot" or nop or noop or "no operation")	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 07:41
S74	2052	("garbage collection" or "garbage collecting") and (state or status or bit or condition or value) and (event or handl$3 or suspend$3 or trigger or trap$3 or exception or suspension)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 16:52
S75	751	("garbage collection" or "garbage collecting") and (state or status or bit or condition or value) and (event or handl$3 or suspend$3 or trigger or trap$3 or exception or suspension) and ("safe point" or consistent)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 16:54
S76	877	("garbage collection" or "garbage collecting") and (state or status or bit or condition or value) and (event or handl$3 or suspend$3 or trigger or trap$3 or exception or suspension) and ("safe point" or consistent or (pointers near3 memory))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 16:55
S77	114	("garbage collection" or "garbage collecting") same (state or status or bit or condition or value) and (event or handl$3 or suspend$3 or trigger or trap$3 or exception or suspension) same ("safe point" or consistent or (pointers near3 memory))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 17:07
S78	2	"6308319".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/02 17:07

S79	12	("5088036" | "5321834" | "5560003" | "5920876" | "5953736" | "6098089" | "6101580").PN. OR ("6308319"). URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/02 17:07
S80	4	("5088036" | "6052699" | "6308319" | "6341293").PN. OR ("6845385").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/02 17:37
S81	7	execut$3 same ("safe point" or (pointers near3 live) or (pointers near3 memory)) same garbage same (suspend$3 or suspension or wait$3 or halt$3)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 07:24
S82	7	execut$3 same ("safe point" or (pointers near3 live) or (pointers near3 memory)) same garbage same (suspend$3 or suspension or wait$3 or halt$3)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 07:24
S83	23	("4558413" | "4807120" | "4922414" | "5535329" | "5590332" | "5764989" | "5835701" | "5987256" | "5995754" | "6151618" | "6206584").PN. OR ("6446257"). URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 07:28
S84	7	("5088036" | "5594904" | "5842016" | "6289360" | "6301704" | "6341293" | "6446257").PN. OR ("6789253"). URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 07:47
S85	1	"6308319".pn.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 07:47
S86	12	("5088036" | "5321834" | "5560003" | "5920876" | "5953736" | "6098089" | "6101580").PN. OR ("6308319"). URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 08:36
S87	0	"determine when to start garbage collector"	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 08:37
S88	2	"start garbage collector"	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 08:38
S89	348	(start$3 or activat$3 or invok$3) near5 (garbage adj collect$3)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 08:39
S90	31	(start$3 or activat$3 or invok$3) near5 (garbage adj collect$3) same (suspend$3 or suspension or mutex or halt$3 or wait$3)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/08/03 08:40

S91	6	"safe point" and (suspen$4 near3 mutator)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/15 09:37
S92	8	("5088036" "5159680" "5941977" "6282633").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/08/15 09:38
S93	141	("5088036").URPN.	USPAT	OR	OFF	2005/08/15 09:43
S94	100	("5088036").URPN. and (parallel or vliw or "delay slot" or "unused position" or (plurality near3 threads))	USPAT	OR	OFF	2005/08/15 09:46
S95	24	("5088036").URPN. and (parallel or vliw or "delay slot" or "unused position" or (plurality near3 threads)) and ("root set" or "safe point" or (pointers near3 garbage))	USPAT	OR	OFF	2005/08/15 10:24
S96	617	(parallel or vliw or "delay slot" or "unused position" or (plurality near3 threads)) and ("root set" or "safe point" or (pointers near4 (memory or heap or garbage))) and (processor near3 (state or status or "condition code" or condition$4 or bit))	USPAT	OR	OFF	2005/08/15 10:28
S97	26	(parallel or vliw or "delay slot" or "unused position" or (plurality near3 threads)) and ("root set" or "safe point" or (pointers near4 (memory or heap or garbage))) same (processor near3 (state or status or "condition code" or condition$4 or bit))	USPAT	OR	OFF	2005/08/15 10:28
S98	32	(parallel or vliw or "delay slot" or "unused position" or (plurality near3 threads)) same ("root set" or "safe point" or (pointers near4 (memory or heap or garbage))) and (processor near3 (state or status or "condition code" or condition$4 or bit))	USPAT	OR	OFF	2005/08/15 10:59
S99	58	S97 or S98	USPAT	OR	OFF	2005/08/15 10:29
S100	1	"5842016".pn.	USPAT	OR	OFF	2005/08/15 10:59

S101	548	707/206.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/18 15:26
S102	359	707/206.ccls. and (mutator or garbage)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/18 15:27
S103	36	707/206.ccls. and (mutator or garbage) and ("root objects" or "safe point" or "coordination points")	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/18 15:29
S104	9	707/206.ccls. and (mutator or garbage) and ("root objects" or "safe point" or "coordination points") and (encod$3 or "unused position" or "delay slot" or vliw)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/18 15:31
S105	34	707/206.ccls. and (mutator or garbage) and ("root objects" or "safe point" or "coordination points") and (state or status or bit or flag)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/18 15:30
S106	580	717/124.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/08 18:15
S107	72	717/124.ccls. and (suspen$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/08 18:15
S108	127	717/124.ccls. and (thread)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/09/08 18:16